第9章

ターゲット機能を実装してパソコン用拡張ボードとして使う

PCIインターフェース回路の製作

井倉将実

ここでは，付属FPGA基板をパソコン用のPCIバスに接続する方法を解説する．FPGAには，PCIインターフェースのターゲット機能を実装する．動作検証を目的として，FPGAが内蔵するメモリ・ブロックへのアクセスや，スイッチによる割り込み発行を行う．（編集部）

付属FPGA基板は小型なものですが，PCIバス・インターフェース機能やメモリ・マップのデバイス・ドライバ開発の支援には十分に活用できます．本稿では，付属FPGA基板をPCIバスに接続する方法を解説します．

筆者が付属FPGA基板をPCIバス・インターフェースに使おうと思いついたのは，

- 長年PCIバス・インターフェースをFPGAで設計してきた
- PCIバス・インターフェースの解説書にHDLコードやデバイス・ドライバなどが提供されている
- PCIバス・インターフェースを持つ組み込み機器が増えている
- PCI関連の開発に手軽に使えるものがほしい

と考えたためです．

設計したPCIバス・インターフェースの機能を**表1**に示します．

表1　設計したPCIバス・インターフェースの機能

PCIコントローラ機能	PCIターゲット機能
データ・バス幅	32ビット・データ・バス
動作クロック	33MHz，または66MHz駆動
バースト転送	非対応．ディスコネクト発行
内蔵機能	8KバイトのSRAMメモリ空間（ベース・アドレス0）
	256バイトのLED/スイッチ/割り込みステータス空間（ベース・アドレス1）
割り込み	1系統．INTA#使用

1．FPGA基板とPCIバスをつなぐ

PCIバスは，4系統の割り込みやバス・マスタのすべてを使用するときには51本，ターゲット機能かつ1系統だけ割り込みを使用する場合は46本のI/Oピンを必要とします．

付属FPGA基板には，2組のユーザ機能拡張コネクタが用意されています．ここにFPGAのユーザI/Oピンのうち，55本が引き出されています．従ってPCIバス・インターフェースが必要とする信号を付属基板でも確保できることになります．

● 付属FPGA基板のI/OをPCIバスに接続する

回路図を**図1**に示します．また，製作した基板を**写真1**に示します．

付属FPGA基板のユーザI/Oのうち5本は，入力専用ピンです．PCIインターフェースのうち，CLK，RST#，IDSEL，PME#，LOCK#は，メイン・ボード側からPCIカード側への信号です．FPGAの入力専用ピンにはこれらの信号を割り当てます．

PME#はFPGAに接続してあるだけで，今回の回路では使用していません．

PME#はOSを介してCPUが休止状態や低消費電力モードに移行するなどの要求を行い，システム全体の電力管理

KeyWord　FPGA，PCI，ターゲット，割り込み，セットアップ・タイミング，DCM，パリティ

図1　付属FPGA基板のI/OをPCIバスに接続する回路

付属FPGA基板のI/OとPCIコネクタをつなぐだけである．

写真1　製作した基板の外観

図2　FPGAに実装する機能
PCIバス・インターフェース部(ターゲット部)，ユーザ・メモリ部，LED/スイッチ制御部を実装した．

を実現するものです．PCIデバイスに対して電力給電を停止するということもあります．FPGAは電源がOFFになると回路情報が消えてしまいます．付属FPGA基板のようにROMが未実装の状態では電源ON時に回路情報の書き込みが必要になるので，電源のON/OFFは現実的ではありません．そのため，FPGAに内蔵したPCIバス・コントローラを制御するためのコンフィグレーション・レジスタには，あらかじめPME機能は「使用しない/対応しない」設定にしています．

LOCK#を用いた排他バス・アクセスは，バス・マスタ機能を考えるときにのみ必要です．ターゲット機能では単純に書き込みと読み出しを行うことだけを考えます．

なお，PCIバス・マスタ機能を実現する際には，REQ#とGNT#の2本の信号が必要になります．今回はターゲット機能の設計でしたが，将来バス・マスタ機能にも対応できるように，これらの信号も接続してあります．

2．PCIインターフェース回路の設計

今回FPGAに実装した回路のブロック図を**図2**に示します．PCIバス・インターフェース部(ターゲット部)，ユーザ・メモリ部，LED/スイッチ制御部を実装しました．

● PCIバス・インターフェース回路の設計

PCIバス・インターフェース部の回路とデバイス・ドライバ，テスト・プログラムは，書籍で解説されていたもの[1]をベースに若干の変更を行いました．LED/スイッチ制御部からバス・インターフェース回路に接続されるスイッチの押下による割り込み機能をサポートするために，コンフィグレーション・レジスタ内に割り込みレベル・レジスタを実装したことと，PCI Rev.2.3以降にコンフィグレーション・レジスタ空間に追加されたレジスタ群を実装したことです．

FPGA内部にはPCIホスト側から自由に読み書きできる8Kバイトのメモリと，256バイトの制御用レジスタを搭載しています(**図3**)．そこで，ベース・アドレス・レジスタ空間は8Kバイトに設定しました．

これらのメモリをPCIバスのメモリ空間に割り当てるため，PCIコントロール・レジスタとしてはメモリ空間有効フラグを実装します．I/O空間有効フラグは実装せず，読み出し時にはゼロを返します．

● メモリ・ブロック機能の実装

32ビット・バス幅の8Kバイト・メモリは，FPGAに内蔵されているメモリ・ブロック(Block RAM)を利用して作りました．

メモリ・ブロックを使用するためのテンプレートは，ISE WebPACKに用意されているため，VHDLやVerilog HDLのソース・コードに組み入れることもできます．

リスト1は，PCIバス・インターフェース回路にメモリ・ブロックを接続する部分のVHDL記述です．PCIバス・インターフェースの回路が32ビット・バス幅でアクセスされるため，メモリも32ビット・バス幅にしています．

しかしPCIデバイスでは，バイト単位のメモリ書き込みも考慮しなければなりません．そこで実際には8ビット・バス幅のメモリを四つ用意して，それぞれのメモリ書き込

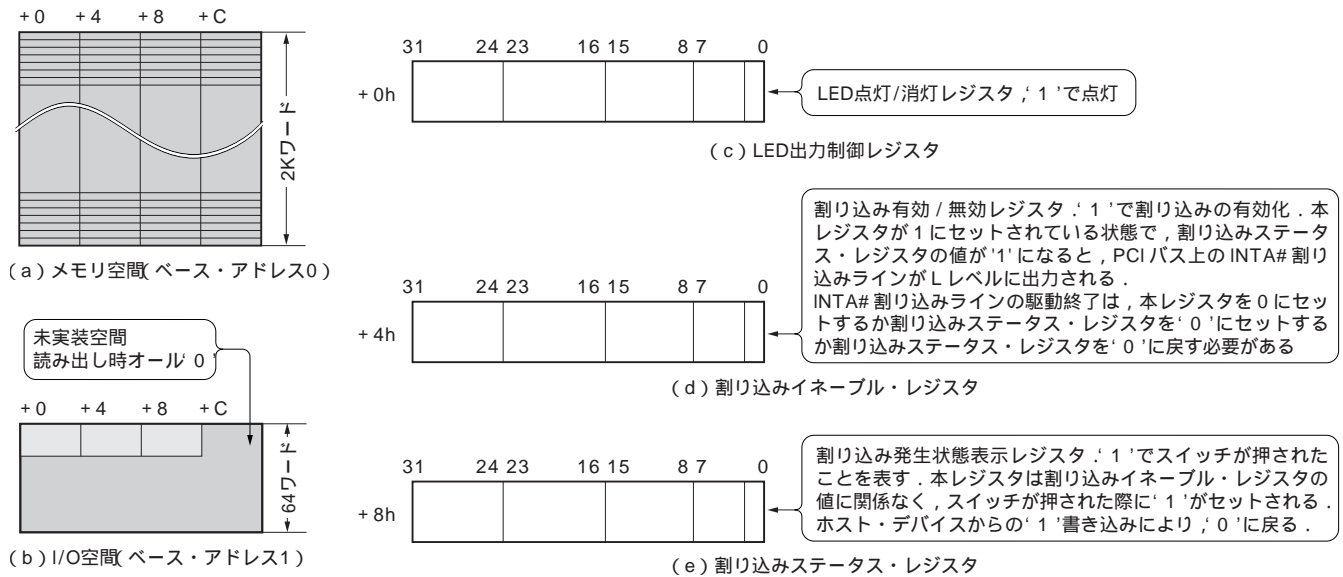



図3　メモリ・マップとレジスタ
FPGA内部にはPCIホスト側から自由に読み書きできる8Kバイトのメモリと，256バイトの制御用レジスタを搭載している．

み信号にはPCIバスのC/BE#[3:0]を使っています(**リスト2**).

メモリの読み出し側はC/BE#[3:0]バイト・イネーブル信号は関係ないので，32ビット単位でデータの読み出しを行っています.

FPGA内蔵メモリは，クロックに同期してデータを読み書きします．そのため，メモリ駆動クロックが必要です．また，読み出し時にアクセス要求を行ってから最低1クロック以上の出力遅延が発生します．今回の設計は性能追及が目的ではないため，PCIバス・クロックをそのまま使用しました.

バス・クロックを2倍か4倍にてい倍して使えば，実質的には遅延がないメモリ・アクセスが可能です(**図4**).

● LED/スイッチ機能

FPGAにはメモリ空間に割り当てているLED出力とスイッチ入力の回路を実装しています．LED出力用に1ビットのレジスタを用意しています.

タクタイル・スイッチの入力は，割り込み信号として使用します．スイッチのチャタリングを考慮したノイズ除去回路を入れてあります．PCIバス・クロックによるスイッチ入力の同期化と，スイッチがOFFになるときの立ち上がりエッジを検出する回路です．動作イメージを**図5**に示します．立ち上がりエッジを検出後，割り込み回路に割り込み要求が発生したことを伝えます.

● 割り込み回路

割り込み回路では，PCI-INTA#信号への割り込み通達を行うか行わないかという指示と，タクタイル・スイッチの押下による割り込みが発生したかどうかを示すレジスタを用意します．前者が割り込みイネーブル・レジスタで，後者が割り込みステータス・レジスタです.

LED/スイッチ機能からは，**図5**で示した立ち上がりのタイミングを検出して，この回路に割り込み要求があったかどうかを通知します．この回路ではこの要求値を保持して割り込みステータス・レジスタに'1'をセットします.

この割り込みステータス・レジスタはCPUから'1'を書き込んでクリアされるか，PCIのバス・リセットが発行されるまで値は保存されます.

割り込みイネーブル・レジスタが'1'にセットされている場合は，PCI-INTA#信号がアサートされます.

割り込みイネーブル・レジスタかステータス・レジスタのいずれかのビットをクリアすることで，INTA#がディアサートされ，割り込み要求クリアとなります.

一般には，割り込みイネーブル・レジスタは'1'にセットし続けたまま，割り込みステータス・レジスタをクリアする方法がとられます.

リスト1
メモリ・ブロックを接続する部分のVHDL記述

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
library UNISIM;
use UNISIM.VComponents.all;

entity PCI_TOP is
      port (

               ～ 中略 ～

      ) ;

end PCI_TOP;

architecture Behavioral of PCI_TOP is

               ～ 中略 ～

component RAM_Block is
      port (
               CLK      : in     std_logic ;
               RST_n    : in     std_logic ;
               ADRS     : in     std_logic_vector(15 downto 2) ;
               BE_n     : in     std_logic_vector( 3 downto 0) ;
               DATA_I   : in     std_logic_vector(31 downto 0) ;
```

リスト2
バイト単位のアクセスを実現するためのVHDL記述

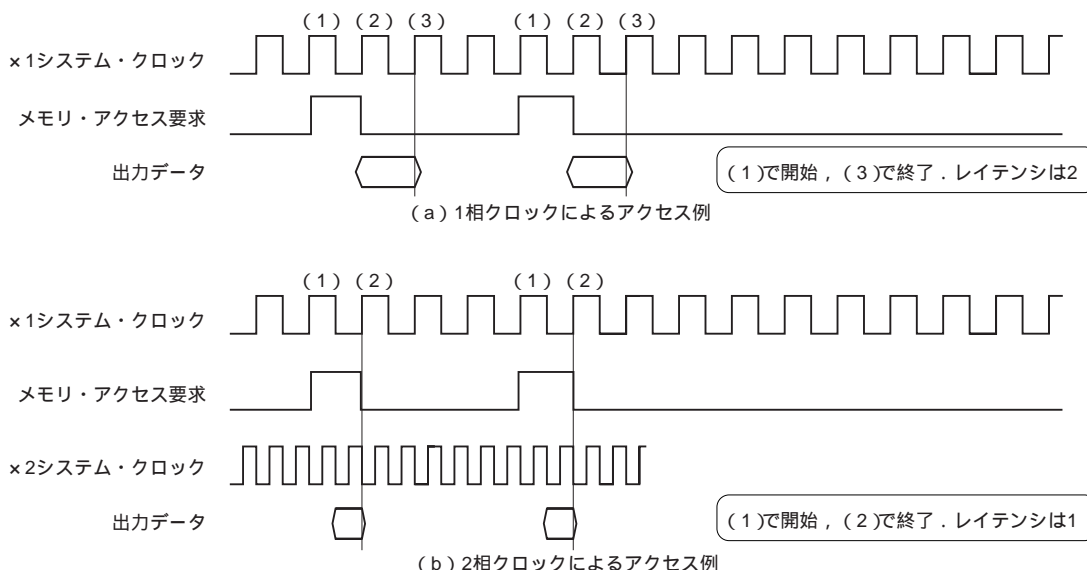
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
library UNISIM;
use UNISIM.VComponents.all;

entity RAM_Block is
      port (
               CLK      : in     std_logic ;
               RST_n    : in     std_logic ;
               ADRS     : in     std_logic_vector(15 downto 2) ;
               BE_n     : in     std_logic_vector( 3 downto 0) ;
               DATA_I   : in     std_logic_vector(31 downto 0) ;
               DATA_O   : out    std_logic_vector(31 downto 0)
      ) ;
end RAM_Block;

architecture Behavioral of RAM_Block is

      component sram_w11_b9 is            // 11ビットアドレス空間，9ビットメモリブロックの宣言
      port (
               clk      : IN     std_logic;
               en       : IN     std_logic;
               sinit    : IN     std_logic;
               addr     : IN     std_logic_VECTOR(10 downto 0);
               din      : IN     std_logic_VECTOR(8 downto 0);
               we       : IN     std_logic;
               dout     : OUT    std_logic_VECTOR(8 downto 0)
      ) ;
      end component ;

               ～ 中略 ～


begin

Control_Equ : process ( CLK )
begin
      if ( CLK'event and CLK = '1' ) then
               SSR      <= not RST_n ;
               EN       <= not SSR ;
      end if ;
end process ;
```

```
                DATA_O    : out     std_logic_vector(31 downto 0)
        ) ;
end component ;

                ～ 中略 ～

begin

                ～ 中略 ～

--**********************************************************************************--
u_RAM_Block : RAM_Block
        port map (
                CLK                 => int_PCICLK ,
                RST_n     => RST_n ,
                ADRS      => MEM_ADRS(15 downto 2) ,
                BE_n      => MEM_WE_n( 3 downto 0) ,
                DATA_I    => MEM_DATA_WrPort ,
                DATA_O    => MEM_DATA_RdPort
        ) ;

        MEM_WE_n(0)         <= '0' when ( MEM_CEn = '0' and MEM_OEn = '1' and MEM_WE0n = '0' ) else '1' ;
        MEM_WE_n(1)         <= '0' when ( MEM_CEn = '0' and MEM_OEn = '1' and MEM_WE1n = '0' ) else '1' ;
        MEM_WE_n(2)         <= '0' when ( MEM_CEn = '0' and MEM_OEn = '1' and MEM_WE2n = '0' ) else '1' ;
        MEM_WE_n(3)         <= '0' when ( MEM_CEn = '0' and MEM_OEn = '1' and MEM_WE3n = '0' ) else '1' ;

                ～ 中略 ～

end Behavioral;
```

```
                ～ 中略 ～

        RAM_Byte0 : sram_w11_b9                        -- D[07:00]ビット目のメモリブロック宣言
        port map (
                CLK       => CLK,                            -- Clock
                sinit     => SSR,                            -- Synchronous Set/Reset Input
                EN        => EN,                             -- RAM Enable Input
                ADDR      => ADRS(12 downto 2),              -- 11-bit Address Input
                din       => WrDATA_Byte0(8 downto 0),       -- 8-bit Data Input
                WE        => WE(0),                          -- Write Enable Input
                dout      => RdDATA_Byte0(8 downto 0)        -- 8-bit Data Output
        );

        RAM_Byte1 : sram_w11_b9                        -- D[15:08]ビット目のメモリブロック宣言
        port map (
                CLK       => CLK,                            -- Clock
                sinit     => SSR,                            -- Synchronous Set/Reset Input
                EN        => EN,                             -- RAM Enable Input
                ADDR      => ADRS(12 downto 2),              -- 11-bit Address Input
                din       => WrDATA_Byte1(8 downto 0),       -- 8-bit Data Input
                WE        => WE(1),                          -- Write Enable Input
                dout      => RdDATA_Byte1(8 downto 0)        -- 8-bit Data Output
        );

        RAM_Byte2 : sram_w11_b9                        -- D[23:16]ビット目のメモリブロック宣言
        port map (
                CLK       => CLK,                            -- Clock
                sinit     => SSR,                            -- Synchronous Set/Reset Input
                EN        => EN,                             -- RAM Enable Input
                ADDR      => ADRS(12 downto 2),              -- 11-bit Address Input
                din       => WrDATA_Byte2(8 downto 0),       -- 8-bit Data Input
                WE        => WE(2),                          -- Write Enable Input
                dout      => RdDATA_Byte2(8 downto 0)        -- 8-bit Data Output
        );

        RAM_Byte3 : sram_w11_b9                        -- D[31:24]ビット目のメモリブロック宣言
        port map (
                CLK       => CLK,                            -- Clock
                sinit     => SSR,                            -- Synchronous Set/Reset Input
                EN        => EN,                             -- RAM Enable Input
                ADDR      => ADRS(12 downto 2),              -- 11-bit Address Input
                din       => WrDATA_Byte3(8 downto 0),       -- 8-bit Data Input
                WE        => WE(3),                          -- Write Enable Input
                dout      => RdDATA_Byte3(8 downto 0)        -- 8-bit Data Output
        );

end Behavioral;
```

(a) 1相クロックによるアクセス例

(b) 2相クロックによるアクセス例

図4 クロックをてい倍してアクセスを高速化する方法
FPGA内蔵メモリは，クロックに同期してデータを読み書きするため，読み出し時にアクセス要求を行ってから最低1クロック以上の出力遅延が発生する．バス・クロックを2倍か4倍にてい倍して使えば，実質的には遅延がないメモリ・アクセスが可能．

● 消費リソースはわずか5%

今回設計したPCIターゲット機能をXC3S250Eに実装した結果，消費したリソースは全体の5%程度でした．9割以上の空き領域は，自由に使用できます．

3. FPGAによるPCIインターフェース設計の利点

FPGAをPCIバス・インターフェース設計に応用して便利になった例を紹介します．

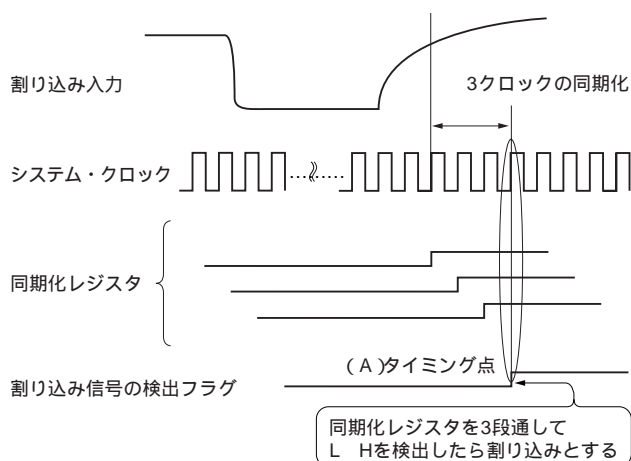



図5 スイッチによる割り込み発行プロセス
スイッチ入力の同期化を行い，スイッチがOFFになるときの立ち上がりエッジを検出する．

● PCIセットアップ・タイミング仕様の確保

付属FPGA基板に搭載のSpartan-3EなどのFPGAは，内部にDCMというクロック管理機能が搭載されています．このDCMをうまく利用すると，PCIバス・インターフェースで規定されている厳しいタイミング仕様を満たしやすくなります．

PCIバス・インターフェース規格Rev2.3では，33MHzと66MHzの2種類のタイミング仕様が規定されています．このうちの66MHzのタイミング仕様であれば，入力信号の最小セットアップ時間はわずかに3nsです．FRAME#やIRDY#のようにPCIホスト・デバイスからの信号をFPGAが受け取って内部で使用するまでに，わずかに3nsしかないという意味になります．

複雑な制御条件が入っているPCIバス規格では，FPGAの入力I/Oパッド内にある同期フリップフロップが使えません(**図6**)．このために，I/Oパッドの遅延とパッドから内部論理セルまでの配線遅延，さらに論理セル＋フリップフロップのセットアップ時間の合計値が，3nsを越えないように設計する必要があります．この数値はFPGAではそうとう厳しい要求です．

DCMを使えば，クロックの位相調整が可能です．256分の1の分解能で，FPGA内部のクロックを遅らせたり進めることが可能です．この機能により．セットアップ時間を確保できるようになります．

今回の設計では，FRAME#信号を使う回路の信号経路

が4.8nsのセットアップ時間を要求しました．1.8nsだけ多くなります．これは，66MHzの1周期を256分割した値と比較すると，約31/256になります．

よって，DCMの設定を以下のように行うことで，1.8nsずれた内部クロックを作り出すことができるようになります．

- CLKOUT_PHASE_SHIFTをFIXEDに設定する
- PHASE_SHIFT値を31に設定する
- CLK0からCLKFBに内部フィードバックする．CLK0出力はそのまま内部の回路駆動用PCIバス・クロックとして使用する

DCMによる位相シフトは，ディジタル制御されたものなので，高い精度で行えます．しかし，入力されるクロックに大きなジッタが存在すると，そのまま位相シフトした結果にも影響を及ぼすことになります．入力されるクロックにはデューティ比が0.5，かつ300ps以下のクロック・ジッタが要求されます．PCIバス・インターフェースの規格に完全準拠したクロック・ソースであれば，この数値は満たしているので，DCMを安心して使用できます．

● パリティ付き内蔵メモリで信頼性を高める

昨今のシステムは動作中における信頼性の確保が重要であり，FPGA内部のメモリといえども正しくデータが保持されていることが重視されます．PCIバス信号にもPAR（パリティ）信号が用意されており，データ・フェーズ中にADバスから1クロック遅れて奇数パリティ信号が出力されます．

付属FPGA基板のSpartan-3Eは，パリティに対応したメモリ・ブロックを持ちます．よって，ホストから出力されたデータを書き込む際には，PCIバス信号のPAR信号と，FPGA内部で計算したパリティ信号を比較して一致していることを確認してからデータとパリティを書き込むことが可能です．

読み出し時にはメモリから読み出したデータとパリティを比較して正しいことを確認してからバスに出力します．

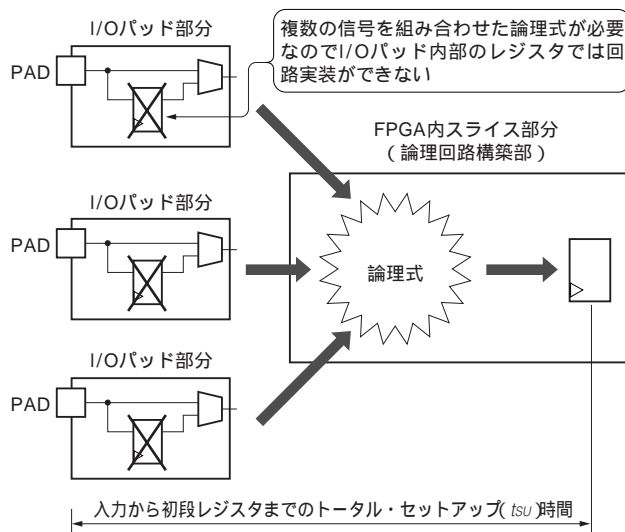



図6 セットアップ時間を確保しにくい理由
複数の信号の組み合わせ論理が必要なので，I/Oパッド内のレジスタを活用できない．

エラーがあった場合には直ちにその状態を把握してPERR#（パリティ・エラー検出）信号をアサートし，システム管理側にエラーを通達します．

高い信頼性が要求されるシステムでは，パリティを活用すべきと筆者は考えます．

参考・引用*文献

(1) Interface編集部編；改訂新版 PCIデバイス設計入門，TECH-I Vol.25，CQ出版社．

いくら・まさみ
アヴネット ジャパン（株）

<筆者プロフィール>
井倉将実．来栖川電工，ひびきのシステムラボ，ライトイメージの取締役を経て，2006年11月にアヴネットジャパンに入社．技術専門誌にPCI，USB，PCMCIAなどのバス・インターフェースやFPGAを応用した製品開発事例の執筆を行う．ここ数年は，日本貿易振興機構（JETRO）とともにタイ/フィリピンへ組み込みシステムスペシャリストとして現地日本企業雇用者向けの技術移転や教育カリキュラムの考案/作成に尽力する．1970年代東京生まれ．